UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/670,898	09/25/2003	William J. Masek	LOT920030024US1	5987

45544 7590 07/28/2011

HOFFMAN WARNICK LLC
75 STATE ST
14TH FLOOR
ALBANY, NY 12207

EXAMINER	
MITCHELL, JASON D	

ART UNIT	PAPER NUMBER
2193	

NOTIFICATION DATE	DELIVERY MODE
07/28/2011	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

PTOCommunications@hoffmanwarnick.com

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/670,898
Filing Date: September 25, 2003
Appellant(s): MASEK ET AL.

Nathan B. Davis (reg. no. 67,474)
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 5/9/11 appealing from the Office action mailed 12/10/10.

(1) Real Party in Interest

The examiner has no comment on the statement, or lack of statement, identifying by name the real party in interest in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The following is a list of claims that are rejected and pending in the application:

Claims 1-5, 7-9, 11-14, 16-18, 20-23 and 25-26 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 6,002,871 to Duggan et al. (Duggan) in view of US 2004/0199815 A1 to Dinker et al. (Dinker) in view of US 7,062755 B2 to Partamian et al. (Partamian) in view of US 5,987,517 to Firth et al. (Firth).

Claims 6, 15 and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 6,002,871 to Duggan et al. (Duggan) in view of US 2004/0199825 A1 to Dinker et al. (Dinker) in view of US 7,062755 B2 to Partamian et al. (Partamian) in view of US 5,987,517 to Firth et al. (Firth) in view of "The Java ^{tm} Virtual Machine Specification" by Lindholm et al (Lindholm).

(4) Status of Amendments After Final

The examiner has no comment on the appellant's statement of the status of amendments after final rejection contained in the brief.

(5) Summary of Claimed Subject Matter

The examiner has no comment on the summary of claimed subject matter contained in the brief.

(6) Grounds of Rejection to be Reviewed on Appeal

The examiner has no comment on the appellant's statement of the grounds of rejection to be reviewed on appeal. Every ground of rejection set forth in the Office action from which the appeal is taken (as modified by any advisory actions) is being maintained by the examiner except for the grounds of rejection (if any) listed under the subheading "WITHDRAWN REJECTIONS." New grounds of rejection (if any) are provided under the subheading "NEW GROUNDS OF REJECTION."

WITHDRAWN REJECTIONS

The following grounds of rejection are not presented for review on appeal because they have been withdrawn by the examiner.

Claims 1-9, 11-18 and 20-26 rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement.

(7) Claims Appendix

The examiner has no comment on the copy of the appealed claims contained in the Appendix to the appellant's brief.

(8) Evidence Relied Upon

6,002,871	Duggan et al.	12-1999
2004/0199825	Dinker et al.	10-2004
7,062,755	Partamian et al.	6-2006

5,987,517 Firth et al. 11-1999

T. Lindholm, F. Yellin "The Java Virtual Machine Specification" 1st de. September 1995.

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claims 1-5, 7-9, 11-14, 16-18, 20-23 and 25-26 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 6,002,871 to Duggan et al. (Duggan) in view of US 2004/0199815 A1 to Dinker et al. (Dinker) in view of US 7,062755 B2 to Partamian et al. (Partamian) in view of US 5,987,517 to Firth et al. (Firth).

Regarding Claims 1, 9 and 18: Duggan discloses:

providing a test application that satisfies reentrancy requirements *(col. 21, lines 57-61 'Each session is … reentrant')* on a client *(col. 5, lines 18-21 'the test tool … runs on a single computer')*;

identifying command modules associated with the test application *(col. 12, lines 21-23 "A list box 272 contains a list of all of the commands in the command module created for testing a given application program", the command module is inherently identified to the list box in order for the list box to present all of the commands from that module; col. 14, lines 22-28 "the command module is implemented as a Visual Basic*

5.0 code module, Each command of the command module comprises a Visual Basic subroutine that contains the instructions for the execution segment of the command");

providing a test script capable of invoking the command modules *(col. 13, lines 59-62 "a test operator [can] create test scripts containing … command module commands")*; and

instantiating, via the test script, a plurality of instances of the test application using threads *(col. 21, lines 57-61 'Each session is executed as a separate thread'; col. 21, lines 46-50 "handles the execution of a test run based on a ... test script")*.

Duggan discloses instantiating and executing a plurality of instances of the test application under the control of a single application (*col. 21, lines 53-61 'The basic module 12 is also responsible for initiating multiple, concurrent sessions … Each session is executed as a separate thread … It is the multi-threaded, reentrant nature of the test tool program code"*). However, Duggan does not explicitly disclose the instantiating and execution of each of the plurality of instances of the test application occur within a single process, without requiring multiple processes to instantiate the plurality of instances within.

Dinker teaches a testing program which instantiates and executes each of a plurality of instances of the test application as one of a plurality of threads in a process (par. [0028] Each test agent 110 may be implemented as a multithreaded application"; par. [0031] "multi-threaded test processes (i.e., test agents 110)").

It would have been obvious to one of ordinary skill in the art at the time the invention was made to instantiate and execute each of the plurality of instances of the test application (Duggan *col. 21, lines 53-61 'initiating multiple, concurrent sessions … Each session is executed as a separate thread"; Dinker par. [0028] "Each test agent 110 may be implemented as a multithreaded application"; par. [0031] "multi-threaded test processes (i.e., test agents 110)")* within a single process without requiring multiple processes to instantiate the plurality of instances within (e.g. by only implementing Duggan's single test agent 110 instead of Dinker's multiple test agents). Those of ordinary skill in the art would have been motivated to do so as one of a finite set of known and implementable methods of providing the disclosed functionality (*i.e. the threads are either implemented in the same process or different processes*) which would produce only the expected results (Duggan *col. 21, lines 53-61 'initiating multiple, concurrent sessions … Each session is executed as a separate thread … It is the multi-threaded, reentrant nature of the test tool program code"; par. [0031] "multi-threaded test processes (i.e., test agents 110)"*).

Duggan discloses the code of the individual threads can safely share services and memory space which are exclusively dedicated to the single process with the other threads (col. 21, lines 53-61 "each session is reentrant"). However, Duggan and Dinker do not explicitly teach sharing all services and memory space which are exclusively dedicated to the single process among the plurality of instances.

Partamian teaches sharing all services and memory space which are exclusively dedicated to the single process among the plurality of instances (*col. 3, lines 12-17 "A process may have one or a plurality of threads. Threads in the same process share information using memory, atomic instructions, mutexes, semaphores, etc."; note that this teaches "Threads in the same process" and does not indicate that threads in a different process have access to the services and memory space; further note that interprocess sharing mechanisms are distinctly described*).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to share all services and memory space (*Partamian col. 3, lines 12-17 "Threads in the same process share information using memory, atomic instructions, mutexes, semaphores, etc."*) among the plurality of threads (*Duggan col. 21, lines 53-61 'initiating multiple, concurrent sessions … Each session is executed as a separate thread … each session is reentrant"; par. [0031] "multi-threaded test processes (i.e., test agents 110)"*). Those of ordinary skill in the art would have been motivated to do so as one of a finite set of known and implementable method of providing the disclosed functionality (i.e. either the threads share memory and services or they don't) which would have resulted in only the expected results (*Duggan col. 21, lines 53-61 'initiating multiple, concurrent sessions … Each session is executed as a separate thread … each session is reentrant"; Partamian col. 3, lines 12-17 "Threads in the same process share information using memory, atomic instructions, mutexes, semaphores, etc."*).

Duggan, Dinker and Partamian do not explicitly teach the command module implemented as APIs.

Firth teaches the use of APIs *(col. 2, lines 63-67 "functions in the Internet API reside in a dynamic link library (DLL)*").

It would have been obvious to one of ordinary skill in the art at the time the invention was made to implement Duggan's command module *(col. 14, lines 22-28 "the command module is implemented as a Visual Basic 5.0 code module)* as an API and to provide a data entry field in the GUI to identify particular API's for use with the application under test. Those of ordinary skill in the art would have been motivated to do so because Firth's APIs "eliminate the need to embed source code directly in an application program to manage Internet application protocols" *(col. 2, lines 64-67; also see Duggan col. 16, lines 9-15 "each command simulates a real user's interaction ... by generating ... an HTTP request")* and thus provide further abstraction for Duggan's test script development *(see e.g. col. 13, lines 59-67 "No knowledge of the underlying programmed instruction of the command module is needed by a test operator"; col. 14, lines 2-4 "The command module is rewritten and/or customized for each different application program to be tested")*.

Regarding Claim 2: The rejection of claim 1 is incorporated; further Duggan discloses; and

upon execution, the test script instantiates the plurality of instances of the test application *(col. 5, line 67-col. 6, line 3 'the test tool program executes multiple concurrent sessions')* using threads *(col. 21, lines 57-61 'Each session is executed as a separate thread')* within the single process *(col. 21, lines 53-57 'The basic module 12 is also responsible for initiating multiple, concurrent sessions'; col. 21, lines 57 "It is the multi-threaded, reentrant nature of the test tool program code")*.

Regarding Claims 3, 14 and 23: The rejection of claims 1, 9 and 18 are incorporated respectively, further; Duggan discloses the server application is a network application *(col. 5, lines 9-12 'a test tool for testing application programs … over a network')*.

Regarding Claims 4, 12 and 21: The rejection of claims 1, 9 and 18 are incorporated respectively, further; Duggan discloses the reentrancy requirements dictates that the plurality of instances of the test application be run within the single process without interfering with each other *(col. 21, lines 57-61 'reentrant nature of the test tool')*.

Regarding Claims 5, 13 and 22: The rejection of claims 1, 9 and 18 are incorporated respectively, further; Duggan discloses each of the plurality of instances of the test application corresponds to a separate thread *(col. 21, lines 57-61 'Each session is executed as a separate thread')*, and wherein each of the separate threads is

associated with a different connection to the server *(col. 5, line 66-col. 6, line 3 'A "session" refers to the execution of one test script, on one client connection')*.

Regarding Claims 7, 16 and 25: The rejection of claims 1, 9 and 18 are incorporated respectively, further; Duggan discloses the plurality of instances of the test application simulate use of the server application by a plurality of users *(col. 6, lines 47-51 'the test tool program … is capable of executing test scripts … based on a user list')*.

Regarding Claims 8, 17 and 26: The method of claim 1, 9 and 18 further comprising collecting data corresponding to the test *(col. 8, lines 4-6 'The test tool program … provides four options for logging information')*.

Regarding Claims 11 and 20: The rejection of claims 9, and 18 are incorporated respectively, further; Duggan discloses, and wherein upon execution, the test script instantiates the plurality of instances of the test application *(col. 5, line 67-col. 6, line 3 'the test tool program executes multiple concurrent sessions')* using threads *(col. 21, lines 57-61 'Each session is executed as a separate thread')* within the single process *(col. 21, lines 53-57 'The basic module 12 is also responsible for initiating multiple, concurrent sessions')*.

Claims 6, 15 and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 6,002,871 to Duggan et al. (Duggan) in view of US 2004/0199825 A1 to Dinker et al. (Dinker) in view of US 7,062755 B2 to Partamian et al. (Partamian) in

view of US 5,987,517 to Firth et al. (Firth) in view of "The Java [tm] Virtual Machine Specification" by Lindholm et al (Lindholm).

Regarding Claims 6, 15 and 24: The rejection of claims 1, 9 and 18 are incorporated respectively, further; Duggan does not disclose the process comprises a JAVA virtual machine.

Lindholm teaches that JAVA programs which run on a JAVA virtual machine were well known at the time of the invention, and that JAVA programs and the JVM provided benefits known to those of ordinary skill in the art.

It would have been obvious to a person of ordinary skill in the art at the time of the invention to implement Duggan's 'test tool' and 'basic module' in the JAVA programming language and execute them on a JVM.

(10) Response to Argument

I. Rejection of claims 1-9, 11-18 and 20-26 under 35 U.S.C. §112, first paragraph.

The 35 U.S.C. §112, first paragraph rejection of these claims has been withdrawn. According no response to the appellants' arguments will be provided.

II. Rejection of claims 1-5, 7-9, 11-14, 16-18, 20-23 and 25-26 under 35 U.S.C. §103(a) as being unpatentable over Duggan, Dinker, Partamian and Firth.

Claim 1

Regarding the limitation:

> … instantiating, via the test script, a plurality of instances of the test application using threads, wherein the instantiating and execution of each of the plurality of instances of the test application occur within a single process …

the appellants state:

> … Appellants respectfully disagree with the combination of the references. For instance, as is clearly stated in Dinker, the test agents 110 are consistently a plurality of test agents, not a single agent. Dinker in fact teaches away from the single test agent of Duggan, stating that "by implementing test cluster 100 from several multi-threaded test processes (i.e. test agents 110)" there is "less thread starvation than if the same number of clients were simulated by a single multi-threaded process." (Dinker, Para. 0031). As such, Dinker viewed alone or in combination with Duggan fails to teach or suggest the feature of instantiating each of the plurality of instances of the test application within one process, and rather teaches directly away from this feature, specifying that a single multi-threaded process would result in thread starvation as cited above. Therefore, there would have been no motivation to combine the teachings of Duggan and Dinker, as Dinker specifically suggests against a single process. Appellants assert that the Examiner is using Appellants' own disclosure as motivation to combine the references. (1st partial par. on pg. 8)

The examiner respectfully disagrees. Duggan discloses a single test object ("basic module 12") spawning a plurality of instances of the test application using threads (col. 21, lines 53-65 "The basic module 12 is also responsible for initiating multiple, concurrent sessions … Each session is executed as a separate thread"). What Duggan does not explicitly state is that this single test object (i.e. "basic module 12") is a process. Those of ordinary skill in the art seeking to implement Duggan's system would necessarily have to provide an implementation which executes either one or a plurality of multi-threaded processes.

Dinker teaches a test object (“test agent 110") which is a single process that instantiates and executes a plurality of instances of the test application using threads (par. [0031] “multi-threaded test processes (i.e., test agents 110)”; par. [0028] Each test agent 110 may be implemented as a multithreaded application").

Those of ordinary skill in the art would have recognized that implementing Duggan’s test object (“basic module 12”) as taught by Dinker’s “test agent 110” (i.e. a single process having multiple threads) would have provided the functionality disclosed in Duggan (i.e. col. 21, lines 53-65 “The basic module 12 is also responsible for initiating multiple, concurrent sessions), and would have done so without the added complexity required to create and manage multiple processes in addition to the multiple threads.

Further, Dinker par. [0031] reads:

> By implementing test cluster 100 from several multi-threaded test processes (i.e., test agents 110, some embodiments may encounter less thread starvation than if the same number of clients were simulated by a single multi-threaded process. Additionally, each test agent may be executed on a lower performance (and thus lower cost) computing device than would be required to execute such a single multi-threaded process.

Note that here Dinker indicates that “*some* embodiments *may* encounter *less* thread starvation”. It is the examiner's understanding that this indicates only that Dinker prefers to implement the testing environment using a plurality of multi-threaded process. Contrary to the appellants' arguments such a preference does not constitute a teaching away from a single multi-threaded process. (see e.g. In re Gurley, 27 F.3d 551, 554, 31 USPQ2d 1130, 1132 (Fed. Cir. 1994). "A known or obvious composition does not become patentable simply because it has been described as somewhat inferior to some other product for the same use."). Particularly it is noted that Dinker is teaching a trade

off which reduces thread starvation at the cost of incorporating additional processes and computing devices. It should be recognized that there are any number of situations where a person of ordinary skill in the art would choose a somewhat higher level of thread starvation over the greater complexity required for additional processes and computing devices.

Finally, the examiner notes that nowhere does Duggan discuss generating multiple processes. Accordingly, it could be argued that because Duggan does not disclose spawning multiple processes, Duggan discloses a method as claimed of "instantiating a plurality of instances of the test application using threads … without requiring multiple processes to instantiate" them. In other words, it is improper to read a disclosure which does not describe any processes as *requiring* multiple processes.

Regarding the limitation:

> … sharing all services and memory space exclusively dedicated to the single process with others of the plurality of instances …

The appellants state:

> … [T]he Examiner cites a passage of Partamian that teaches a general description of processes and threads, specifically citing that "threads in the same process share information using memory, atomic structures, mutexes, semaphores, etc." (Partamian, Col. 3, lines 12-17, emphasis added). Applicants, however, assert that the Examiner has misinterpreted the passage cited, as emphasized above. The passage states that threads within a process share information by using memory, not that they share a memory space, or even that there is a shared memory at all. There is further no reference to sharing services. Appellants further assert that there is no disclosure that the services and memory space are dedicated exclusively for the single process. As such, Partamian clearly fails to teach or suggest this feature, as sharing information using memory clearly is not equivalent to services and memory space dedicated for a single process. …
> (par. bridging pp. 8 and 9)

The examiner respectfully disagrees. First it is noted that the claimed limitation is very broad. In particular the term “all services and memory space exclusively dedicated to the single process” is at best broadly defined because, neither the claim nor the specification describes what memory space or services are exclusively dedicated to the single process or how this exclusivity is determined/enforced.

Partamian discloses one set of tools or methods used to share information among threads in the same process (e.g. col. 3, lines 13-15 "Threads in the same process share information using memory, atomic instructions, mutexes, semaphores, etc."), and another set of tools or methods used to share information between different processes (e.g. col. 3, lines 15-17 "processes share information using file system, sockets, shared memory, semaphores, dynamic data exchange, etc."). This disclosure is understood to describe tools or methods exclusively dedicated to sharing data between the threads of a single process (i.e. "memory, atomic instructions [and] semaphores").

Further, the appellants have not presented a distinction between threads that “share information by using memory” and “share a memory space”. It is noted that information stored in memory takes up space in that memory. Accordingly, two threads that share information using memory are sharing the memory space used to store that information. Accordingly the examiner does not perceive a distinction between the cited Partamian and the limitation in question.

Similarly, the appellants have not presented a distinction between the cited mutexes and semaphores and the claimed services. The disclosed mutexes and

semaphores fall within a reasonably broad interpretation of the scope of a "service". In particular, these mutexes and semaphores are services provided by the operating environment and used and shared by the threads thus teaching the claimed limitation.

Alternatively, it should be noted that the appellants have argued:

> … Since the testing is specifically based on the load generated, it would be clear to one of ordinary skill in the art that the services and memory are dedicated to this testing, or the results would not be conclusive. To this extent, Appellants assert that these portions of the specification, inter alia, implicitly and/or inherently support the claimed limitation objected to by the Examiner. …
> (par. bridging pp. 6 and 7)

Duggan discloses a system in which the testing is specifically based on the load generated (see e.g. col. 21, lines 59-62 "It is the multi-threaded, reentrant nature of the test tool program code that allows multiple sessions to execute concurrently, enabling a test operator to stress the application program under test to almost any degree"[1]). Accordingly, it would seem that the claimed exclusivity would also be implicitly and/or inherently disclosed by Duggan.

Regarding the limitation reciting:

> … identifying application protocol interfaces APIs …, prior to the instantiating step …[and] providing a test script capable of invoking the APIs ...

The appellants state:

> … As a first example, this reference describing API simply describes what an API is with no reference to test applications or any related information. Further, as Appellants have repeatedly argued, the Examiner attempts to replace whole pages of Duggan that describe the formation of scripts with one generic sentence describing where an API is stored. Appellants submit that this combination is faulty

[1] Note that in the context of Duggan stress and load testing are considered synonymous (see e.g. col. 1, lines 36-39 "Subjecting an application program to access by multiple clients … is generally referred to as load or stress testing".

> as Duggan's entire disclosure relies upon the use of Visual Basic 5.0. For instance, Col. 14 of Duggan using Visual Basic is not the only reference to the program, rather the entire description of code up to Col. 21 is a description of the code used with Visual Basic. This description is preceded by Col. 13 of Duggan describing that the Test Tool Program also uses Visual Basic, and that the code modules and forms are the building blocks of the program. (Duggan, Col. 13, Lines 22-41). As such, entire pages of Duggan describe the use of Visual Basic, which is the Microsoft Program used in the entirety of the application. In fact, at Page 11 of the Final Office Action, the Examiner quotes the ease of use of Duggan with no knowledge of underlying programmed instruction of the command module needed, citing Col. 14, Lines 2-4 of Duggan. Appellants note that this is in reference to the preferred embodiment described in both Col. 13 and Col. 14, which utilizes the Visual Basic program for such an ease of use. Applicants respectfully submit that the reference to an API in Firth has nothing to do with creating testing of application programs, as in Duggan, and any attempt to incorporate this generic API into the Visual Basic GUI based system of Duggan would in fact, at best, lead to undue experimentation and yield unpredictable results. In fact, the Examiner argues that a generic teaching that APIs exist is sufficient to show that implementing Duggan's test environment in APIs was obvious, quoting which file formats to use. (Final Office Action, Page 5). Appellants assert that the Examiner is using personal knowledge rather than relying upon prior art, and the rejection is thus faulty. Accordingly, Appellants request that the rejection of claim 1 be reversed.

The examiner respectfully disagrees. First, it is noted that the appellants have repeatedly referred to “Visual Basic” as a “program”. Specifically it should be noted that "visual basic" is not a “program", it is a “programming language” used to write programs. More specifically, it is noted that Visual Basic is a programming language capable of creating and calling to APIs and thus does not preclude the use of APIs (see e.g. Frith col. 20, lines 50-52 “the Internet API functions can be used from programming languages (e.g. Visual Basic® by Microsoft)”). Accordingly it is not clear what the appellants mean when they assert “the Examiner attempts to replace whole pages of Duggan". In fact it seems more accurate to say that, based on the teachings of Firth, the examiner is attempting to replace references to Duggan’s “command module” with

references to a “command API”. This is no more than a change in how the collection of commands are implemented and would not affect the overall functionality of Duggan.

More specifically Duggan discloses a “command module” which is a collection of routines callable from other modules (see e.g. col. 14, lines 22-28 "Each command of the command module comprises a Visual Basic subroutine that contains the instruction for the execution segment of the command"). Duggan’s command module and associated functions provide functionality for creating and testing application programs (see e.g. col. 12, lines 21-23 “A list box 272 contains a list of all of the commands in the command module created for testing a given application program”).

Firth teaches APIs as a method to provide a collection of reentrant routines for access by external programs (col. 2, lines 49-52 “An internet application program interface (API) … including a set of reentrant Internet-specific functions”). Accordingly, those of ordinary skill in the art would have understood that Frith’s API constitutes a known alternative method of providing the functionality of Duggan’s command module (e.g. a command API).

Alternatively, it is noted that while Duggan consistently refers to his “command module” as a “module” (and not explicitly as an API), the functionality provided by the command module is, at least to a significant extent, equivalent to the functionality provided by an application programming interface. For example, the Microsoft Computer Dictionary 5th ed. provides the following definition for an API:

> **application programming interface** n. A set of routines used by an application program to direct the performance of procedures by the computer’s operating system.

Duggan's command module is used by an application program (col. 14, lines 35-36 "The Encode__Command is called … by the core module") to direct the performance of procedures by the computer operating system (col. 14, lines 23-26 "Each command of the command module comprises a Visual Basic subroutine that contains the instructions for the execution segment of the command"). Thus it could be argued that, given a reasonably broad interpretation, Duggan's command module constitutes the claimed API.

Claims 2-5 and 7-9, 11-14, 16-18, 20-23 and 25-26

The appellants' arguments regarding these claims refer to and rely on the arguments addressed above and thus stand or fall with claim 1.

III. Rejection of claims 6, 15 and 24 as being unpatentable over Duggan in view of Dinker, Partamian, Firth and Lindholm.

Claims 6, 15 and 24

The appellants' arguments in this section merely assert that Lindholm does not cure the asserted deficiencies addressed above. Accordingly these claims also stand or fall with claim 1.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

/Jason Mitchell/

Primary Examiner, Art Unit 2193

Conferees:

/Lewis A. Bullock, Jr./

Supervisory Patent Examiner, Art Unit 2193

/Emerson C Puente/

Supervisory Patent Examiner, Art Unit 2196